ISBN: 978-628-96613-1-6. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2025.1.1.1253

# Comparison Analysis of Convolutional Neural Networks Using CPU and GPU in the Diagnosis of Lung Diseases

Lisset Fernanda Maldonado Lezama[1]; Eder Omar Moreano Rojas[2]; Pedro Huamaní-Navarrete[3]
[1,2,3]Ricardo Palma University, Perú, *lisset.maldonado@urp.edu.pe, eder.moreano@urp.edu.pe, phuamani@urp.edu.pe*

***Abstract*– This article describes the comparative analysis of CPU and GPU usage in training and validating three convolutional neural network models, to diagnose lung diseases from central chest X-rays. Since tuberculosis and Covid-19 present similar symptoms, it is possible to confuse the diagnosis and provide incorrect treatment. For this reason, two convolutional network models, InceptionV3 and VGG16, and an arbitrary one consisting of ten hidden layers, were chosen to compare them and thus achieve a more accurate diagnosis. Likewise, the JupyterLab interface was used with the Python programming language, complemented by the TensorFlow and Keras libraries. Then, for the training stage, 2,535 images were used, and the transfer learning technique was applied using the computer's CPU and GPU, with the purpose of analyzing the effectiveness when comparing each of the presented cases; As well as, among the types of patterns to be recognized, the diagnosis of healthy patients was included. Regarding the evaluation, the metrics Accuracy, Recall, F1-Score, and General Precision were used to identify the performance of the arbitrary network model when compared to the other two models mentioned. In this way, the arbitrarily proposed convolutional neural network model achieved higher accuracy, equivalent to 92.70% when the CPU was used, while when the GPU was used, this accuracy increased to 94.28%.***

***Keywords*-- InceptionV3 model, VGG16 model, CPU, GPU, Transfer learning, frontal chest x-ray.***

# Análisis de comparación de redes neuronales convolucionales utilizando CPU y GPU en el diagnóstico de enfermedades pulmonares

Lisset Fernanda Maldonado Lezama[1]; Eder Omar Moreano Rojas[2]; Pedro Huamaní-Navarrete[3]
[1,2,3]Ricardo Palma University, Perú, *lisset.maldonado@urp.edu.pe, eder.moreano@urp.edu.pe, phuamani@urp.edu.pe*

***Resumen*– Este artículo describe el análisis de comparación en el uso de un CPU y GPU en el entrenamiento y validación de tres modelos de redes neuronales convolucionales, para diagnosticar enfermedades pulmonares a partir de radiografías de tórax central. Pues, debido a que la tuberculosis y el Covid-19 presentan síntomas similares, es posible confundir el diagnóstico y otorgar un tratamiento incorrecto. Por tal razón, se eligieron dos modelos de redes convolucionales InceptionV3, VGG16, y uno arbitrario conformado por diez capas ocultas, para compararlos y así lograr un diagnóstico más acertado. Asimismo, se recurrió a la interfaz JupyterLab mediante el lenguaje de programación Python, complementado con las librerías Tensorflow y Keras. Luego, para la etapa de entrenamiento se utilizaron 2535 imágenes, y se recurrió a la técnica de transfer learning utilizando el CPU y GPU del computador, con el propósito de analizar la eficacia al comparar cada uno de los casos presentados; así como también, entre las clases de patrones a reconocer, se incluyó el diagnóstico de pacientes sanos. Respecto a la evaluación, se utilizaron las métricas Accuracy, Recall, F1-Score y General Precision, para identificar el desempeño del modelo de red arbitraria al compararlo con los otros dos modelos mencionados. De esta manera, el modelo de red neuronal convolucional propuesto arbitrariamente obtuvo una mayor precisión y equivalente a 92.70% cuando se utilizó el CPU, mientras que cuando se empleó el GPU dicha precisión se incrementó a 94.28%.***

***Palabras claves*-- Modelo InceptionV3, Modelo VGG16, CPU, GPU, Transfer learning, radiografía de tórax frontal.***

## I. Introducción

A consecuencia de la pandemia del Covid-19 aún existen personas que persisten con las enfermedades pulmonares, lo cual origina que los médicos de los centros de salud manifiesten cierta dificultad en la atención rápida de los pacientes para otorgar un diagnóstico temprano; así como también, dicho aprieto se debe a que las enfermedades pulmonares Tuberculosis y Covid-19 presentan síntomas iniciales y similares tales como la tos, la fiebre, los escalofríos, entre otros síntomas más. Por esa razón, es muy común solicitar al paciente una serie de exámenes médicos como es la radiografía de tórax para diagnosticar el tipo de enfermedad. Sin embargo, estas dos enfermedades pueden ser confundidas y puede otorgarse un tratamiento incorrecto, lo cual podría llevar al deceso del paciente. Por lo cual, en esta investigación se planteó como objetivo analizar tres modelos de redes neuronales convolucionales utilizando la CPU (unidad central de procesamiento) y GPU (unidad de procesamiento gráfico) de una computadora portátil estándar, para diagnosticar enfermedades pulmonares a partir de imágenes digitalizadas de radiografías de tórax frontal, lo cual apoyaría al personal médico en el diagnóstico rápido y preciso para posteriormente otorgar el adecuado tratamiento a tiempo. La implementación fue desarrollada en la interfaz JupyterLab con apoyo de las librerías de Tensorflow y Keras, usando el lenguaje de programación Python.

No obstante, en la literatura se han encontrado diversas aplicaciones de estas arquitecturas de redes neuronales utilizando distintas plataformas de trabajo, variadas enfermedades, lenguajes de programación y bases de datos. Es así como, en [1] se evaluaron los modelos de redes neuronales convolucionales Resnet50, Inceptionv3 y uno particular, para apoyar el proceso de diagnóstico de neumonía asociada al Covid-19; por lo cual, después de modificar los modelos, la arquitectura InceptionV3 alcanzó la mejor exactitud con un valor de 0.9886.

Luego, en [2] se utilizaron los modelos MobileNet y ResNet modificados para la clasificación de dos conjuntos de imágenes, con la finalidad de resolver el problema de desaparición del gradiente y mejorar el rendimiento de la clasificación, mediante la combinación dinámica de características en las diferentes capas, logrando mejores resultados en cuanto a exactitud, sensibilidad y precisión de clasificación. Asimismo, según [3], se recomienda el uso de transfer learning en Python y con las librerías Keras y Tensorflow, para el entrenamiento del modelo InceptionV3 y la clasificación de patrones asociados a enfermedades en distintas imágenes médicas, tales como resonancias magnéticas, tomografías y radiografías del tórax de personas adultas.

Posteriormente en [4], se indicaron dos repositorios de datos disponibles públicamente que contienen imágenes de rayos X de tórax sanas e infectadas, los cuales fueron utilizados para la detección del Covid-19 en imágenes de rayos X de tórax, realizando el ajuste de los pesos de algunas de las capas superiores.

De igual forma, según [5] el primer paso realizado fue el preprocesamiento de las imágenes para eliminar el ruido, seguido de la extracción de características profundas utilizando múltiples modelos profundos previamente entrenados; luego, se realizó la clasificación a partir de las características obtenidas para decidir si el paciente está infectado por coronavirus o se trata de otra enfermedad

pulmonar. Paralelamente, según [6], el uso de los modelos de redes neuronales profundas con aprendizaje por transferencia, VGG19 y U-Net, permitieron la clasificación del Covid-19 en diagnóstico positivo/acertado o negativo/equivocado; sin embargo, fue necesario una etapa de preprocesamiento con segmentación pulmonar, para eliminar el entorno que no ofrece información relevante, seguido de un análisis e interpretación mediante mapas de calor.

Por otro lado, en [7] se propuso un modelo de aprendizaje profundo, DarkNet, para la clasificación binaria de Covid-19 y una del tipo multiclase (Covid-19, no hallado y neumonía), a partir de imágenes de rayos X de bases de datos públicas, utilizando 17 capas convolucionales y diferentes filtros por capa. Y, según [8] el uso del modelo de red LSTM y un marco de optimización de características para la clasificación de Covid-19, en imágenes públicas, permitió el análisis de la fusión de fuentes múltiples y funciones redundantes; para ello, también fue necesario aumentar el contraste de las imágenes utilizando una combinación de algoritmos de filtrado, seguido de la extracción de características.

## II. Marco Teórico

En esta sección se abordan ciertas definiciones referentes al desarrollo de este trabajo de investigación, el cual fue producto de una sustentación de tesis para la obtención del título de Ingeniero Electrónico, en la Universidad Ricardo Palma, Lima, Perú [9].

### A. *Red neuronal convolucional*

Según [10], es un modelo de arquitectura compuesta por una capa de entrada, una capa de salida y muchas capas intermedias ocultas. Estas capas realizan operaciones que alteran los datos con el objetivo de aprender características específicas de los mismos. Además, las tres capas más frecuentes son las de convolución, activación o ReLU, y pooling. A continuación, en la figura 1 se muestra la arquitectura general de una red neuronal convolucional.

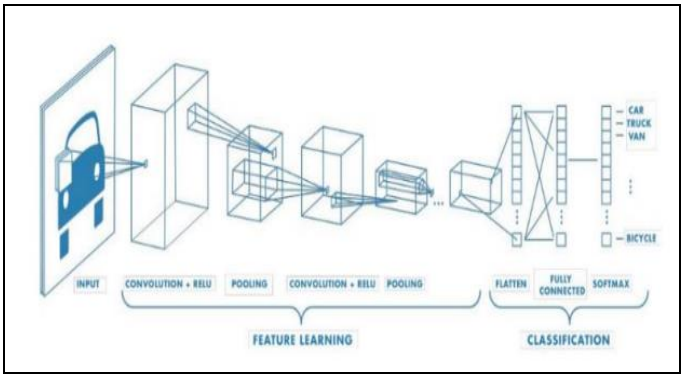


Fig. 1. Arquitectura general de una red neuronal convolucional [10]

### B. *Arquitectura de red neuronal InceptionV3*

Según [11] es un modelo de red neuronal que fue introducida en el año 2015 con el nombre de GoogleNet. Este modelo de red contiene 48 capas y estas mismas pueden clasificar hasta 1000 clases distintas. Tiene como un tamaño máximo de entrada de 299 x 299, un pooling, otra convolución de 1x1 y al final una convolución de 1x1. La salida de este módulo termina siendo una concatenación de cuatro ramas, siendo la salida del módulo base la concatenación de estas. A continuación, en el lado izquierdo de la figura 2 se muestra la arquitectura de este modelo de red neuronal.

### C. *Arquitectura de red neuronal VGG16*

Visual Geometry Group (VGG) inventó el VGG-16 que tiene 16 capas convolucionales y 3 completamente conectadas, lo cual lleva consigo la parte tradicional de ReLu (AlexNet). La red VGG-16 apila muchas más capas en AlexNet y usa filtros con un tamaño menor (2x2 y 3x3); asimismo, esta red tiene más de 138 millones de parámetros y ocupa más de 500MB de almacenamiento [12]. A continuación, en el lado derecho de la figura 2 se muestra la arquitectura de este modelo de red neuronal.

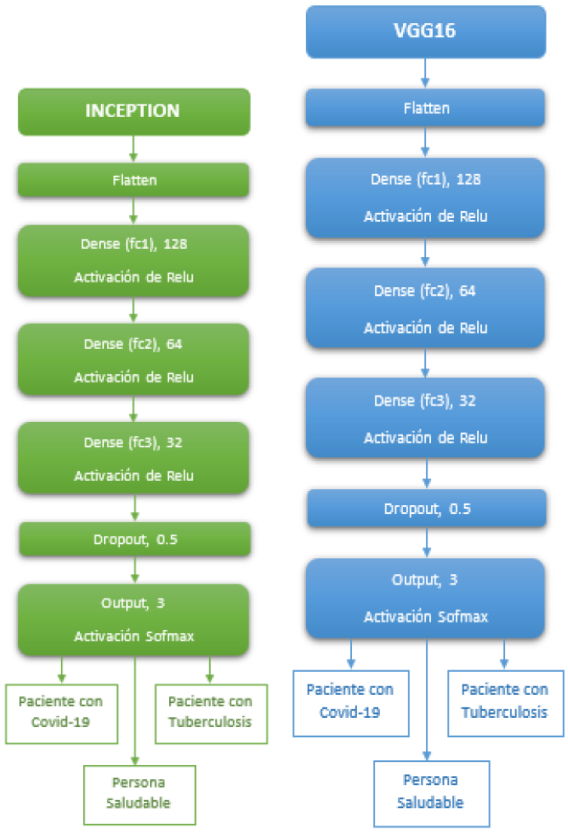


Fig. 2. Izquierda: arquitectura del modelo de red neuronal convolucional InceptionV3. Derecha: arquitectura del modelo de red neuronal convolucional VGG16.

## III. Tratamiento y Evaluación de las Imágenes del Dataset

El DataSet se limitó a un total de 2535 imágenes y correctamente balanceadas para las tres situaciones

propuestas: personas con tuberculosis, con Covid-19 y completamente sanas; además, fueron extraídas de bases de datos de IEEE [13].

### *A. Tratamiento y evaluación de las imágenes*

Para el tratamiento y evaluación de las imágenes de la base de datos se procedió a dividirlas en 3 grupos: con tuberculosis, con Covid-19 y sanas. Luego, se realizó la conversión de todas las imágenes del modelo de color RGB a formato de escala de grises, usando librerías en Python desde la Plataforma JupyterLab, tal como I.convert('L'). Con las imágenes en escalas grises, se procedió a calcular el promedio de píxeles de cada imagen para determinar aquellas con buena o mala calidad.

### *B. Evaluación y selección de las imágenes*

Los datos otorgados por el algoritmo desarrollado fueron trasladados a un archivo de Excel, para luego realizar un histograma y conocer la ubicación del mejor promedio de píxeles. En la figura 3 se muestra el histograma logrado luego de retirar las imágenes que presentaron un mejor promedio de píxeles, las cuales fueron usadas para el entrenamiento de los modelos de redes neuronales. Por lo cual, las imágenes que se encontraron en los rangos mostrados en el histograma pasaron a ser eliminadas ya que no presentaron una calidad óptima para el entrenamiento.

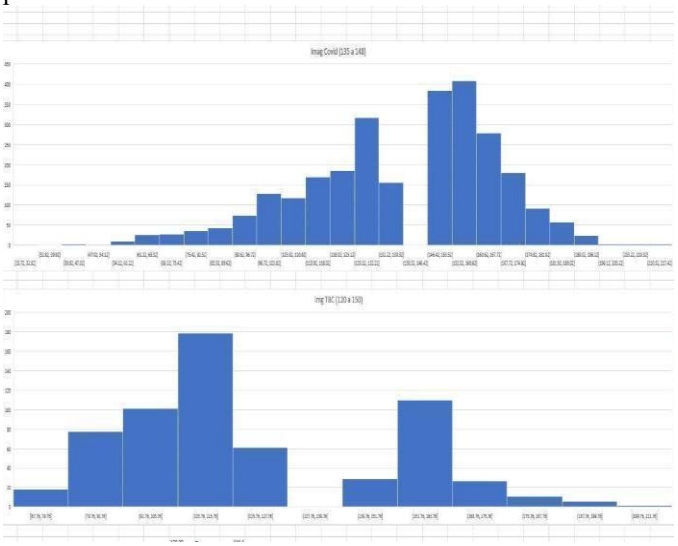

Fig. 3. Histograma de las imágenes descartadas por prormedio de pixeles.

Asimismo, se decidió seleccionar varias imágenes con buena calidad por la cual se obtuvo el rango de promedio de pixels adecuado: 80, 120 y 150, de los cuales se concluyó que las imágenes en el rango de 120 a 140 son imágenes que presentan mayor calidad para el entrenamiento de las arquitecturas de redes neuronales convolucionales propuestas.

Para la eliminación de las imágenes, se usó el Excel y el Command prompt debido a que las imágenes por descartar no fueron correlativas. Fue así como se utilizó la función "=TRANSPONER" en el archivo en Excel, el cual permitió pasar los datos de la columna a una celda en la cual se mostraron concatenados y separados por coma.

Luego de finalizar todo este proceso, se obtuvo un total de 2535 imágenes para los tres grupos: Covid-19, Tuberculosis y Personas Sanas. Donde el 80% (676 imágenes) de cada grupo se almacenó en una carpeta denominada Train, y el 20% en una carpeta Val (169 imágenes de validación para cada grupo), y un 20% restante de Val y Train se almacenó en una carpeta llamada Testeo para su posterior uso.

### *C. Preprocesamiento de las imágenes*

Para el procesamiento se procedió a definir los hiper parámetros que se usaron para la etapa del entrenamiento, en el caso de épocas fue igual a 120 y correspondió a la cantidad de veces que se iteraron para su entrenamiento, y referente a "batch size" fue igual a 64 y representó la cantidad de imágenes que ingresaron en cada "paso". Además, en el preprocesamiento se necesitó importar la librería de Tensorflow instalado en su versión 2.1 (para el uso del GPU) ya que en la API de Keras (versión 2.4.3) solo disponía de la versión 2.5. Adicionalmente, para su uso se necesitó tener instalado CUDA 11.3 de NVIDIA y cuDNN 8.2.

De igual forma, en la implementación de los modelos de redes neuronales se necesitó de Keras ya que este incluye librerías para la implementación y entrenamiento. Asimismo, se optó por usar la librería ImageDataGenerator para realizar el preprocesamiento de imágenes, ya que permitió dividir y generar el proceso de entrenamiento en batchs (bloques), además de que realiza la técnica data augmentation. En cuanto al parámetro Rescale se le asignó 1./255 que indica que las imágenes que están en el directorio train_datagen se normalizan, Shear_range = 0.3 para generar imágenes con inclinación y así la red aprenda a trabajar con imágenes que no están bien encuadradas, Horizontal_flip = True lo cual permite invertir la imagen en forma horizontal, y Zoom_range = 0.3.

Luego de indicar los parámetros, estos fueron usados para definir las variables "train_generator" y "val_generator", las cuales contienen las imágenes preprocesadas con los parámetros indicados previamente. Dentro de los cuales se declaró un "target_size", el cual permitió redimensionar el tamaño de las imágenes a una especificada. En la figura 4 se muestra parte del algoritmo utilizado para el entrenamiento y validación de las imágenes del modelo particular.

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical')

val_generator = val_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical')
```

```
Found 2028 images belonging to 3 classes.
Found 507 images belonging to 3 classes.
```

Fig. 4. Algoritmo con parámetros para el entrenamiento y validación de la red neuronal convolucional.

## IV. Implementación y Entrenamiento de los Tres Modelos de Redes Convolucionales

A continuación, se describe el procedimiento de entrenamiento de las tres arquitecturas de redes neuronales convolucionales, utilizando la CPU y la GPU de la computadora portátil.

### *A. Arquitectura y entrenamiento. Red neuronal arbitraria*

Para el diseño se consideraron 10 capas basado en el método de prueba y error, los cuales fueron analizados para corregir, agregar o quitar ciertas capas de la red logrando implementar una red arbitraria óptima. Asimismo, se tomó como referencia el modelo de red implementada en [3], y considerando lo realizado en [14] en cuanto a la aplicación de filtros para afinar la red respecto a la extracción de características de las imágenes, también al uso de la activación softmax para la capa de salida permitiendo clasificar las 3 clases descritas y así minimizar las dimensiones en el proceso de la red. De igual forma, para reducir la posibilidad de sufrir un sobre ajuste se tomó en consideración lo realizado en [15] al utilizar la capa Dropout para el proceso de entrenamiento de los modelos de redes neuronales convolucionales. Seguidamente, se describen las 10 capas de este modelo, desde el preprocesamiento hasta la salida multiclase.

- La primera capa de convolución tuvo 32 filtros kernels con un tamaño de 4x4, un input shape de 224x224 y la función de activación ReLU. Por lo cual, se obtuvo un total de 1'605, 632 neuronas para la primera capa.
- La segunda capa de convolución tuvo 64 filtros kernels, con un tamaño de 3x3 y se usó la función de activación ReLU. Por lo cual, se obtuvo un total de 3'211,264 neuronas.
- La tercera capa fue un Max Pooling de 2x2, esto permitió reducir la altura y ancho dando una reducción en las dimensiones de la imagen, originando una redimensión a 112x112, lo cual redujo la cantidad de neuronas a 802,816.
- La cuarta capa fue una convolución con 128 filtros de 2x2 y con la función de activación ReLU. Con esto se obtuvieron 128 matrices de salidas.
- La quinta capa fue un Max pooling, de 2x2, esto permitió reducir las dimensiones a un tamaño de 56x56, lo cual redujo la cantidad de neuronas a 401,408.
- La sexta capa fue un Flatten para "aplanar" las dimensiones de las matrices, esto permitió transformar los resultados de tres dimensiones a una dimensión y conservar la misma cantidad de neuronas de la capa anterior.
- La séptima capa fue una Dense de 128 neuronas, para unir todas las neuronas de la capa anterior con esta capa. Esta capa tuvo la activación ReLU como las anteriores.
- La octava capa tuvo un Dropout=0.5, permitiendo desactivar el 50% de todas las neuronas recibidas con el fin de que la red convolucional no "memorice" y que realice un correcto aprendizaje.
- La novena capa fue una "Dense" de 64 neuronas para conectar todas las neuronas de capas anteriores con esta misma, la cual tuvo la función de activación ReLU.
- La última capa tuvo la activación Softmax, que asigna a cada una de las tres clases un valor probabilístico. Por lo cual, la salida con un mayor valor fue el resultado de la evaluación del modelo.

Luego de haber implementado la arquitectura del modelo de red arbitrario, se ejecutó usando la función model.compile(). Esta permite compilar la red por optimizadores, funciones, métricas, etc. Para este caso se usó la función de optimización "Loss" de tipo categorical ("Categorical_crossentropy"), el optimizador usado fue Adam y para las métricas de evaluación se usó "Accuracy".

Por otro lado, para el caso del entrenamiento, se realizó lo siguiente cuando se utilizaron la CPU y GPU.

A.1. Entrenamiento con CPU. Se definieron los hiperparámetros como épocas, bath_size y pasos, los cuales fueron calculados por la cantidad de imágenes de entrenamiento (2028) entre el bath_size que indicó 31 pasos por época. Luego, se ingresó la cantidad de pasos de validación, la cual se obtiene dividiendo el val_Sample entre el bath_size, para observar el aprendizaje de la red. Finalmente se ejecutó el entrenamiento utilizando la función "Fit". Tal entrenamiento por cada época duró 130 segundos y el tiempo total de entrenamiento usando la CPU fue aproximadamente de 4 horas 30 minutos.

A.2. Entrenamiento con GPU. Se instaló inicialmente la librería tensorflow-gpu, luego se definieron los mismos parámetros de entrenamiento del caso anterior, y para el proceso de validación se utilizaron los mismos comandos. El entrenamiento usando la GPU de la tarjeta gráfica de la computadora portátil, se realizó en 24 segundos por época y el tiempo total de entrenamiento fue de aproximadamente 1 hora con 20 minutos.

A continuación, en la figura 5 se muestran los resultados de los datos obtenidos por el Loss y Accuracy del grupo de entrenamiento.

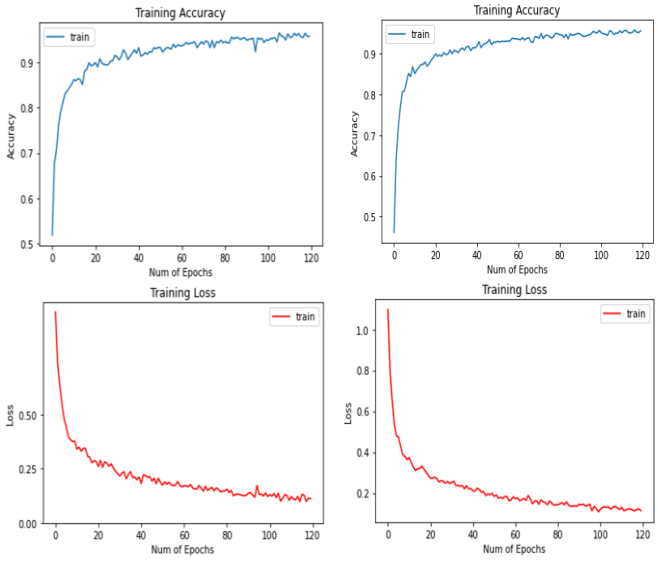


Fig. 5. Gráficos de Accuracy y Loss del entrenamiento de la red neuronal particular. Izquierda con CPU y Derecha con GPU.

*B. Arquitectura y entrenamiento. Red neuronal InceptionV3*

Según [10], este modelo tiene 48 capas en su estructura, de las cuales todas se encuentran pre entrenadas, y está diseñada para admitir hasta 1000 clases con una máxima dimensión de entrada permitida igual a 299x299.

Tal es así que, como primer paso, se aplicó la técnica Transfer Learning para aprovechar el preentrenamiento que posee gracias a la base de datos de ImageNet, la cual se realiza cuando se declaran los argumentos de implementación del modelo de esta red. De esta manera, se añadieron 6 capas al final, y se retiró la capa de predicción de 1,000 clases con el comando "include_top=false". Para esto se tomó como referencia lo realizado en [14] donde se adicionaron 3 capas a la arquitectura: una Pooling, una Dense de 1,024 neuronas y una Softmax, no otorgando resultados eficientes. Por esa razón, se añadieron 6 capas: tres Dense, una Flatten, una Dropout y una de Predicción de 3 salidas. A continuación, se detallan:

- Primero se agregó una capa "Flatten" para llevar las dimensiones de la data recibida a una sola dimensión con las mismas neuronas que su capa anterior.
- Como segunda capa se agregó una Dense que contiene 128 neuronas la cual cuenta con activación ReLU, para transformar los valores negativos en ceros, y al igual que el caso anterior conserva las mismas neuronas que la capa anterior.
- Luego, se agregó una capa Dense que contiene 64 neuronas la cual cuenta con activación ReLU, y al igual que el caso anterior conserva las mismas neuronas que la capa precedente.
- Seguidamente, se agregó otra capa Dense" que contiene 32 neuronas la cual cuenta con activación ReLU, y al igual que el caso anterior conserva las mismas neuronas que la capa precedente.
- Asimismo, se agregó una capa Dropout de 0.5 lo cual hace que el 50% de las neuronas se inhabiliten con el objetivo de que la red no memorice sino más bien que aprenda.
- Y, finalmente, se agregó una capa de salida la cual contiene la activación Softmax, esta asigna a cada una de las tres clases un valor probabilístico. Esta evaluación indica si la imagen pertenece a la clase "Positivo a Tuberculosis", "Positivo a Covid-19" o "Persona Sana"; por lo cual, esta función "Softmax" fue la más ideal para este trabajo porque permitió la clasificación para un caso multiclases.

Y, para el caso del entrenamiento, se realizaron los siguientes pasos cuando se utilizó la CPU y GPU.

*B.1. Entrenamiento con CPU.* Se utilizaron los mismos parámetros mencionados para el entrenamiento de la red arbitraria en la sección anterior, además se usaron la mismas "épocas", "steps_per_epoch", "validation_steps" y "batch_size". Para este caso el modelo fue almacenado con el nombre de "inceptionv3.h5", y se pudo observar que cada época tuvo una duración entre 133 a 140 segundos lo cual hizo que tenga un tiempo de demora aproximado de 5 horas.

*B.2. Entrenamiento con GPU.* Se procedió a instalar la librería tensorflow-gpu la cual permitió usar la GPU como medio de entrenamiento; y, se validó con los comandos ya explicados para el caso de la red neuronal arbitraria. Luego de haber realizado estos pasos, se procedió a definir los mismos parámetros de entrenamiento de la red neuronal con CPU, pero para este caso se almacenó el modelo con el nombre de "inceptionv3.h5". Luego de haberse realizado el entrenamiento se observó que para este caso cada época tuvo una duración de 36 segundos, dando así una duración total de entrenamiento de 1 hora con 12 minutos.

A continuación, en la figura 6 se muestran los resultados de los datos obtenidos por el Loss y Accuracy del grupo de entrenamiento.

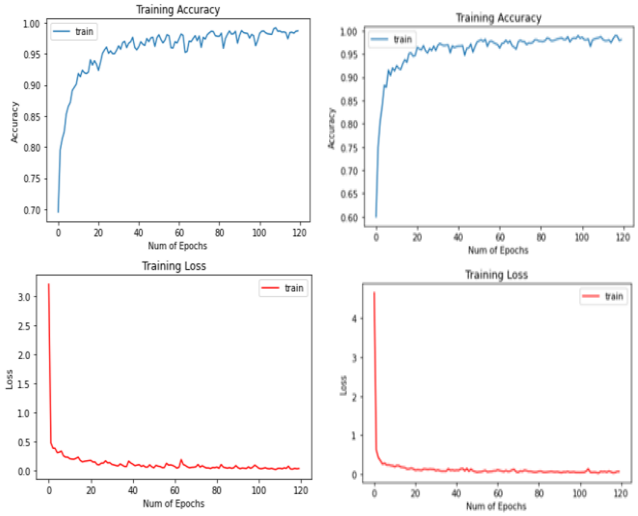


Fig. 6. Gráficos de Accuracy y Loss del entrenamiento de la red neuronal InceptionV3. Izquierda con CPU y Derecha con GPU.

*C. Arquitectura y entrenamiento. Red neuronal VGG16*

De acuerdo con lo indicado en [17], este modelo presenta 16 capas en su estructura, de las cuales todas se encuentran pre entrenadas. Además, está diseñado para admitir hasta 1,000 clases y la máxima dimensión de entrada permitida es 224x224. Y, según [16], esta cuenta con 138'357,544 parámetros entrenados. Por lo cual, al ser este modelo una red pre entrenada con una base de datos de ImageNet se procedió a eliminar la capa de predicción de 1,000 salidas con el comando "include_top=false", para proceder con la aplicación de la técnica de Transfer Learning.

De esta manera, se procedió a utilizar los mismos parámetros mencionados anteriormente en la red arbitraria; por lo cual, para el caso del entrenamiento, se realizaron los siguientes pasos cuando se utilizó la CPU y GPU.

*C.1. Entrenamiento con CPU.* Para el proceso de entrenamiento de la VGG16 se utilizaron los mismos parámetros mencionados para el entrenamiento de la red arbitraria, en la sección anterior. Para ello, se utilizaron las

mismas “épocas”, “steps_per_epoch”, “validation_steps” y “batch_size”. Además, para este caso, el modelo fue almacenado con el nombre de “CPUVGG16.h5”; por lo tanto, se observó que cada época tuvo una duración entre 285 a 434 segundos teniendo un promedio de 359.5 segundos por época, lo cual hizo que tenga un tiempo de entrenamiento de aproximadamente 12 horas.

*C.2. Entrenamiento con GPU*. Para el entrenamiento con la GPU, se procedió a instalar la librería tensorflow-gpu la cual permitió usar la GPU como medio de entrenamiento; asimismo, se validaron los comandos ya explicados para el caso de red arbitraria. Luego de haber realizado estos pasos se procedió a definir los mismos parámetros de entrenamiento de la red con CPU, pero para este caso se procedió a almacenar el modelo con el nombre de “VGG16.h5”. Concluido el entrenamiento, se observó que para este caso con GPU cada época tuvo una duración de 23 segundos, dando así una duración total de entrenamiento de 46 minutos.

A continuación, en la figura 7 se muestran los resultados de los datos obtenidos por el Loss y Accuracy del grupo de entrenamiento.

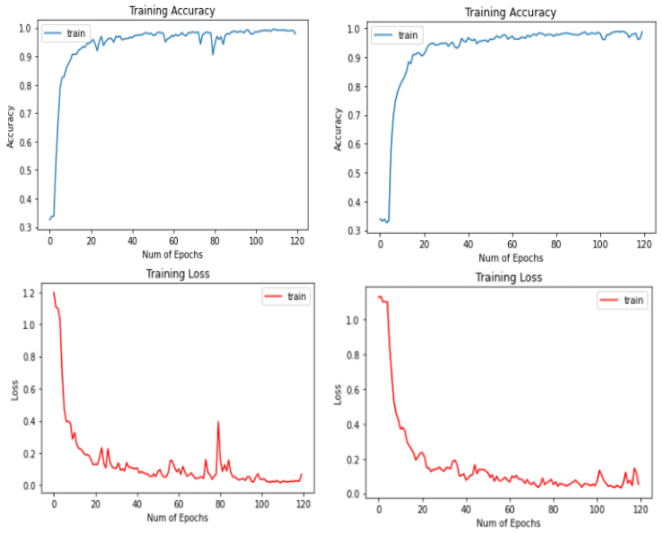


Fig. 7. Gráficos de Accuracy y Loss del entrenamiento de la red neuronal VGG16. Izquierda con CPU y Derecha con GPU.

## IV. Resultados

Seguidamente, en las tablas I y II se muestran los resultados del uso de las métricas Accuracy, Precision, Recall, F1 Score y General Precision, después del entrenamiento de la red neuronal arbitraria propuesta al emplear la CPU y GPU del computador portátil. Por lo cual, se concluye que el uso del GPU fue óptimo.

Respecto a los resultados del modelo de red InceptionV3, las tablas III y IV muestran el Accuracy, Precision, Recall, F1 Score y General Precision, permitiendo de esta manera conocer la precisión de dicho modelo para las tres clases designadas, después de realizar el entrenamiento usando la CPU y GPU.

TABLA I
MÉTRICAS DE MEDICIÓN DEL MODELO DE RED ARBITRARIA UTILIZANDO CPU

| Clases | Métricas | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | Precision | Recall | Specifity | F1-Score | General Precision |
| Covid-19 | 0.9428 | 0.9605 | 0.8639 | 0.9822 | 0.9097 | 0.9270 |
| Sano | 0.9724 | 0.9874 | 0.9290 | 0.9941 | 0.9573 | |
| Tuberculosis | 0.9389 | 0.8520 | 0.9882 | 0.9142 | 0.9151 | |

TABLA II
MÉTRICAS DE MEDICIÓN DEL MODELO DE RED ARBITRARIA UTILIZANDO GPU

| Clases | Métricas | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | Precision | Recall | Specifity | F1-Score | General Precision |
| Covid-19 | 0.9586 | 0.9512 | 0.9231 | 0.9763 | 0.9369 | 0.9428 |
| Sano | 0.9783 | 0.9938 | 0.9408 | 0.9970 | 0.9666 | |
| Tuberculosis | 0.9487 | 0.8907 | 0.9645 | 0.9408 | 0.9261 | |

TABLA III
MÉTRICAS DE MEDICIÓN DEL MODELO INCEPTIONV3 UTILIZANDO CPU

| Clases | Métricas | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | Precision | Recall | Specifity | F1-Score | General Precision |
| Covid-19 | 0.9546 | 0.9932 | 0.8698 | 0.9970 | 0.9274 | 0.9408 |
| Sano | 0.9862 | 0.9939 | 0.9645 | 0.9970 | 0.9790 | |
| Tuberculosis | 0.9408 | 0.8564 | 0.9882 | 0.9172 | 0.9176 | |

TABLA IV
MÉTRICAS DE MEDICIÓN DEL MODELO INCEPTIONV3 UTILIZANDO GPU

| Clases | Métricas | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | Precision | Recall | Specifity | F1-Score | General Precision |
| Covid-19 | 0.8264 | 0.8932 | 0.5444 | 0.9675 | 0.6765 | 0.7968 |
| Sano | 0.9606 | 0.9806 | 0.8994 | 0.9911 | 0.9383 | |
| Tuberculosis | 0.8067 | 0.6426 | 0.9467 | 0.7367 | 0.7656 | |

Y, respecto a los resultados del modelo de red VGG16, las tablas V y VI muestran el Accuracy, Precision, Recall, F1 Score y General Precision, permitiendo de esta manera conocer la precisión de dicho modelo para las tres clases designadas, después de realizar el entrenamiento usando la CPU y GPU.

TABLA V
MÉTRICAS DE MEDICIÓN DEL MODELO VGG16 UTILIZANDO CPU

| Clases | Métricas | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | Precision | Recall | Specifity | F1-Score | General Precision |
| Covid-19 | 0.9665 | 0.9750 | 0.9231 | 0.9882 | 0.9483 | 0.9527 |
| Sano | 0.9822 | 0.9819 | 0.9645 | 0.9911 | 0.9731 | |
| Tuberculosis | 0.9566 | 0.9061 | 0.9704 | 0.9497 | 0.9371 | |

De igual forma, se realizó la comparación entre los tres modelos de redes neuronales convolucionales para visualizar el desempeño de cada una, y así conocer el mejor rendimiento. A continuación, en la tabla VII se observa la comparación de los tres modelos de redes neuronales con cada una de sus métricas de evaluación.

TABLA VI
MÉTRICAS DE MEDICIÓN DEL MODELO VGG16 UTILIZANDO GPU

| Clases | Métricas | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | Precision | Recall | Specifity | F1-Score | General Precision |
| Covid-19 | 0.9625 | 0.9688 | 0.9172 | 0.9852 | 0.9422 | 0.9448 |
| Sano | 0.9744 | 1.0000 | 0.9231 | 1.0000 | 0.9600 | |
| Tuberculosis | 0.9527 | 0.8796 | 0.9941 | 0.9320 | 0.9333 | |

TABLA VII
COMPARACIÓN DE MÉTRICAS EN LOS TRES MODELOS DE REDES NEURONALES

| Modelos de redes neuronales | | Métricas | | | | |
|---|---|---|---|---|---|---|
| | | Accuracy | Precision | Recall | Specifity | F1-Score |
| ARBITRARIA | CPU | 0.9513 | 0.9333 | 0.9270 | 0.9635 | 0.9273 |
| | GPU | 0.9619 | 0.9452 | 0.9428 | 0.9714 | 0.9432 |
| INCEPTION V3 | CPU | 0.9606 | 0.9479 | 0.9408 | 0.9704 | 0.9413 |
| | GPU | 0.8646 | 0.8388 | 0.7968 | 0.8984 | 0.7934 |
| VGG16 | CPU | 0.9684 | 0.9543 | 0.9527 | 0.9763 | 0.9529 |
| | GPU | 0.9632 | 0.9494 | 0.9448 | 0.9724 | 0.9452 |

## V. CONCLUSIONES

Con el procedimiento realizado, se logró diseñar una red neuronal arbitraria con 10 capas e implementado en el lenguaje de programación Python con librerías de Tensorflow y Keras, asimismo la etapa de entrenamiento se realizó con una CPU y GPU de un computador portátil. Para el caso de la CPU se obtuvo un 92.70% de precisión general y un 94.28% para el caso de la GPU, tal como se apreció en las tablas I y II.

De igual forma, se optó por aplicar la técnica de transfer learning a los dos modelos de redes, InceptionV3 y VGG16, para entrenarlas con las imágenes digitalizadas de radiografía de tórax frontal; por lo cual, durante dicha etapa de entrenamiento para el modelo de red neuronal InceptionV3 y usando el CPU se obtuvo una precisión general de 94.08%, mientras que para el caso de la GPU un 79.68%; esto se observa en las tablas III y IV. Por otro lado, para el modelo de red neuronal VGG16, haciendo uso del CPU se obtuvo una precisión general de 95.27% en cuanto al utilizar la GPU se alcanzó un 94.48%, tal como se muestra en las tablas V y VI.

Por último, se logró evaluar los tres modelos de redes neuronales convolucionales, de los cuales se determinó que el mejor rendimiento y con menor sobreajuste para el entrenamiento correspondió a la red neuronal arbitraria; así como también, el uso de la GPU aceleró significativamente el proceso de entrenamiento en comparación al uso del CPU del computador portátil.

## REFERENCIAS


[1] Caya Pérez, J. C. (2020). Evaluación de Modelos de Redes Neuronales Convolucionales aplicado a Radiografías de Tórax, para apoyar al proceso de diagnóstico de Neumonía asociada al Covid-19. Retrieved from https://repositorio.urp.edu.pe/bitstream/handle/URP/3523/ELEC-T030_46733086_T%20%20%20CAYA%20P%C3%89REZ%20JHAN%20CARLOS.pdf?sequence=1&isAllowed=y

[2] Guangyu, J. (2021, 1 julio). Classification of COVID-19 chest X-Ray and CT images using a type of dynamic CNN modification method. ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S0010482521002195

[3] Colula, S., & Ángel, L. (2019). Reconocimiento de patrones en imágenes médicas por medio de redes neuronales convolucionales. (Master's thesis). Benemérita Universidad Autónoma de Puebla, Puebla.

[4] Mohammad Shorfuzzaman, Mehedi Masud, "On the Detection of COVID-19 from Chest X-Ray Images Using CNN-Based Transfer Learning ," *Journal Tech Science Press*, vol. 64, no. 3, June, pp. 1359-1381, 2020.

[5] Hariri, W., Narin, A. "Deep Neural Networks for COVID-19 Detection and Diagnosis Using Images and Acoustic-based Techniques: a Recent Review," *Soft Comput*, vol. 25, pp. 15345-15362, 2021. https://doi.org/10.1007/s00500-021-06137-x

[6] D. Arias-Garzón et al, "COVID-19 Detection in X-Ray Images Using Convolutional Neural Networks," *Journal Machine Learning with Applications*, vol. 6, August, pp. 1-10, 2021.

[7] T. Ozturk et al, "Automated detection of COVID-19 cases using deep neural networks with X-ray images," Journal Computers in Biology and Medicine, vol.121, April, pp. 1-11, 2020.

[8] A. Hamza et al, "COVID-19 Classification Using Chest X-Ray Images: A Framework of CNN-LSTM and Improved max Value moth Flame Optimization," *Journal Frontiers*, August, pp. 1-20, 2022.

[9] L. Maldonado y E. Moreano, "Diseño y aplicación de redes neuronales convolucionales para el diagnóstico de enfermedades pulmonares a partir de radiografías de tórax frontal", Tesis para Título Profesional de Ingeniero, Universidad Ricardo Palma, Lima, Perú, 2021.

[10] Mathworks. (2020). Mathworks. Recuperado el 12 de octubre de 2020, de Redes Neuronales Convolucionales: https://la.mathworks.com/discovery/convolutional-neural-network.html

[11] Gómez-Ríos, A., Tabik, S., Luengo, J., & Herrera, F. "Redes Neuronales Convolucionales para Una Clasificación Precisa de Imágenes de Corales". *In XVIII Conferencia de la Asociación Española para la Inteligencia Artificial*, 1171- 1176, 2019.

[12] N. Urbano, H. Lacomi y M. Lavorato, "B-VGG16: Red Neuronal de Convolución Cuantizada Binariamente para la Clasificación de Imágenes", *Revista electron*, vol. 6, no. 2, pp. 107-114, 2022.

[13] Tawsifur Rahman, Amith Khandakar, Muhammad Enamul Hoque Chowdhury. (2020). Tuberculosis (TB) Chest X-ray Database. IEEE Dataport. https://dx.doi.org/10.21227/mps8-kb56

[14] Narin, A., Kaya, C., & Pamuk, Z, "Automatic Detection of Coronavirus Disease (COVID-19) Using X-Ray Images and Deep Convolutional Neural Networks, *Pattern Analysis and Applications*: PAA, 24(3), 1–14, 2020.

[15] Sharma, H., Jain, J. S., Bansal, P., & Gupta, S., "Feature Extraction and Classification of Chest X-Ray Images Using CNN to Detect Pneumonia", *in 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 227-231, January 2020.

[16] Keras. (2021). Keras documentation: Keras Applications. Recuperado el 01 de Septiembre del 2021, de Keras.io: https://keras.io/api/applications

[17] Yusuf, K. (2020) Ship Detection and Classification using type of Convolution Neural Networks (Master of Science) http://earsiv.cankaya.edu.tr:8080/xmlui/bitstream/handle/20.500.12416/4898/Thesis.pdf?sequence=1&isAllowed=y